DOCUMENT RESUME

ED 025 751 AL 001 634

By-Reich, Peter A.
Symbols, Relations, and Structural Complexity.
Yale Univ., New Haven, Conn. Linguistic Automation Project.
Pub Date May 68
Note-43p.
EDRS Price MF-$0.25 HC-$2.25
Descriptors-Algorithms, *Context Free Grammar, Linguistic Theory, Networks, *Phrase Structure, *Structural Analysis, *Syntax
Identifiers-*Relational Network Approach, Stratificational Grammar

This paper discusses an alternate formalism for context-free phrase structure grammar. The author feels that if a grammar is stated completely explicitly it can be represented in the form of a relational network of the type proposed by Lamb. He discusses some formal properties of such networks and makes some revisions to Lamb's formulation which allow the formal properties and the structural complexity count to be kept as simple as possible. The notion of "hill climbing" on equivalence spaces defined by the formal properties and the complexity count is introduced as a model of the simplification part of language acquisition. The author suggests that using the complexity count he proposes, the process of simplification in disjunctive form is "non-uphill all the way." Thus he finds that the network approach not only gives a simpler overall system, but one which has the added advantage of starting linguists along the path of developing a detailed model of the process of language acquisition. (See related document AL 001 636.) (Author/DO)

SYMBOLS, RELATIONS, AND STRUCTURAL COMPLEXITY

Peter A. Reich

May 1968

Linguistic Automation Project

YALE UNIVERSITY

New Haven, Connecticut

SYMBOLS, RELATIONS, AND STRUCTURAL COMPLEXITY

Peter A. Reich

Modern linguists often describe their data with a system of rules of the form:

a _is_ _rewritten_ _as_ b _in_ _the_ _environment_ _of_ c___d

where a and b are strings of symbols. These symbols may represent a tree structure collapsed into a string by means of phrase markers (labelled parentheses). Within the tree structure the symbols may stand for columns of plusses and minuses. The rules are linearly ordered, or perhaps quasilinearly ordered (Chomsky, 1967; Chomsky and Halle, 1968). This means that in producing a particular utterance, one applies, or attempts to apply, rules to the string one is building up one at a time, in linear sequence. After applying all of the rules, or perhaps a subset of them, one may have to reapply the rules or a subset of them again and again until no further changes can be made to the string. If at the end of this process the string consists solely of terminal symbols, one has produced a grammatical utterance. Basically such systems consist of symbols and a few operations on these symbols, including match, copy, concatenate, and replace. This system grew out of a union of the item-and-process approach to linguistics with automata theory, an area of mathematics, which grew out of the development of computers. This system can be shown to be equivalent to a type of computer called a Turing machine (Markov, 1954), and certain more limited systems can be shown to be equivalent to more limited types of automata (Chomsky, 1963b).

In sharp contrast to this approach is the theory developed by Lamb (1966a, 1966b), in which the system underlying natural language behavior is formalized as a network of logical elements, or relation-ships, which communicate with one another using a small set of dis-crete signals. There are no rules in a stratificational grammar, nor are there symbols in the usual sense of the term. One can, of course, describe the network of relationships in terms of a set of formulas consisting of symbols, which stand for lines in the net-work, and operators, which stand for nodes in the network. In fact we do this in order to input networks to the computer. However the basic form is the network form. The first operation performed by the Relational Network Simulator (Reich, 1968b) is conversion of the formulas back into networks. One insight of Lamb's formulation is that the use of symbols and rules specifying operations on these symbols is not necessary to the description of the system underlying natural language data. This insight is important, because it brings us a small step closer to understanding how the system underlying language might be stored and used in the brain (Reich, 1968a).

Recently published criticism (Chomsky, 1967; Postal, 1968) of Lamb's theory shows considerable misunderstanding of his basically different system.[1] This is one of a series of papers designed to alleviate this misunderstanding by clarifying the relational network approach. The concern of this paper is with the relational network equivalent of context-free phrase structure grammars (Chomsky and Schützenburger, 1963). We shall show that if such grammars are made completely explicit, they are equivalent to symmetric list structures, and are representable as network diagrams. We shall discuss structural complexity in this framework, and consider various formal properties of the relations defined. We shall conclude with a discussion of the implications of these properties for language acquisition.

Consider the grammar given in figure 1. The arrow stands for 'rewrite as', and a space between two symbols represents concatenation.

Initial symbol: S

(1.1) $S \rightarrow NP\ VP$

(1.2) $NP \rightarrow A\ \ N$

(1.3) $VP \rightarrow V\ \ NP$

(1.4) $A \rightarrow$ a

(1.5) $A \rightarrow$ the

(1.6) $N \rightarrow$ boy

(1.7) $N \rightarrow$ girl

(1.8) $V \rightarrow$ hit

(1.9) $V \rightarrow$ kissed

Figure 1: A simple context-free phrase structure grammar

Thus rule 1.1 can be read, 'S is rewritten as NP followed by VP'. These are considered unordered rules (the initial numbers are inserted merely for reference). Depending on the order the rules are applied, any of eight different sentences result. Thus if the rules are applied in the order (1.1 1.2 1.4 1.6 1.3 1.8 1.2 1.5 1.7) the result is 'a boy hit the girl', whereas if they are applied in the order (1.1 1.2 1.5 1.7 1.3 1.8 1.2 1.4 1.6) the result is 'the girl hit a boy'. I suggest that the concept of ordering as described above represents a confusion of two concepts - ordering and disjunction.

When we suggest that these rules are unordered, we are not saying all we know about the sequence in which these rules must be applied. We know that rule 1.1 must be the first rule executed. We know that rule 1.2 must be executed after rule 1.1, and that rule 1.3 must also follow rule 1.1. We know that after (not necessarily immediately after) rule 1.2 is executed we must execute either rule 1.4 or rule 1.5, and we must also execute either rule 1.6 or 1.7. We know that every occurrence of the execution of rule 1.2 must have been preceeded (not necessarily immediately preceeded) either by 1.1 or by 1.3. And so on. What we are doing, of course, is looking at the relationship of the symbols on the right side of the rules to the symbols on the left side.

Consider the occurrence of the symbol A in rule 1.2. There are two occurrences of A on the left side of rules - in 1.4 and in 1.5. Thus we can say that rule 1.2 will be followed by either rule 1.4 or rule 1.5. If we subscript occurrences of each symbol in the grammar as shown in figure 2, we can express this statement as an additional rule, namely rule 2.10, in which the comma stands for the operation of disjuction. We can produce similar rules for the symbols N and V as given in 2.11 and 2.12 respectively. The distribution of NP is just

(2.1) $S \rightarrow NP_1 \quad VP_1$

(2.2) $NP_1 \rightarrow A_1 \quad N_1$

(2.3) $VP_2 \rightarrow V_1 \quad NP_3$

(2.4) $A_2 \rightarrow$ a

(2.5) $A_3 \rightarrow$ the

(2.6) $N_2 \rightarrow$ boy

(2.7) $N_3 \rightarrow$ girl

(2.8) $V_2 \rightarrow$ hit

(2.9) $V_3 \rightarrow$ kissed

(2.10) $A_1 \rightarrow A_2, \quad A_3$

(2.11) $N_1 \rightarrow N_2, \quad N_3$

(2.12) $V_1 \rightarrow V_2, \quad V_2$

(2.13) $NP_1, \quad NP_3 \rightarrow NP_2$

(2.14) $VP_1 \rightarrow VP_2$

Figure 2: The C-F grammar of figure 1 in explicit form

the reverse of the distribution of A, N, and V. In the case of NP there is one occurrence on the left side of a rule (in 2.2) and there are two occurrences on the right side of rules (in 2.1 and 2.3). Here we can say that every occurrence of the execution of rule 1.2 must have been preceeded either by 1.1 or by 1.3. Again we have disjunction, but this time in the other direction. We express this fact in rule 2.13. In the case of VP there is one occurrence of the symbol on both the right and the left sides, so we simply indicate this in rule 2.14.

Rules 2.10 through 2.14 make explicit the implicit ordering in the grammar of figure 1. Obviously we have the same relationship in rules 2.10 through 2.12 as we have in rule 2.13, namely disjunction, but with direction reversed. One of the counterintuitive artifacts of transformational theory is that (following earlier practice in mathematics and logic) a notation was developed to explicitly express disjunction of the type given in rule 2.10, but none was developed to express disjunction of the type given in rule 2.13.[2] In the transformational framework, rules 1.4 and 1.5 would be written $A \rightarrow a$, the. However, disjunction does not seem to have the same status as the operation of concatenation. Rather it is simply an 'abbreviatative notation' (Bach, 1964:17) which represents two separate rules (Chomsky, 1957:110; 1963a:288). It does, however, seem to have higher status than the use of symbols like [p] as abbreviations for the appropriate column of distinctive features (Chomsky, 1965:213 footnote 14). In the case of the former, the abbreviated rule is used in the count of symbols as a measure of complexity (Chomsky, 1965:42ff), whereas in the latter case, one must substitute the distinctive features before counting symbols (see Householder, 1965:16-26).

The purpose of calling disjunction an abbreviation seems to be to keep the system mathematically simple. This reasoning is beyond me. If disjunction is in the system, whether implicitly or explicitly, it is there, and giving it the epithet 'abbreviation' does not make it go away. Since one of the purposes of building a formal system is to make the relationships as explicit as possible, the transformational notation may be said to be deficient with respect to disjunction of the type given in rule 2.13. This lack of explicitness seems to have been vaguely perceived by Bach (1964:51-52) when he wrote, 'It is annoying to have to skim through a long lexical list which turns out not to have the symbol you are looking for, and then to be forced to look through numerous rules to see what happens to the item (sometimes hidden in a complex set of contexts)....It would be worthwhile practice to list the transformations in which a general symbol is mentioned at the first introduction of the item.' The correction of this deficiency has certain rewards, which we shall now explore.

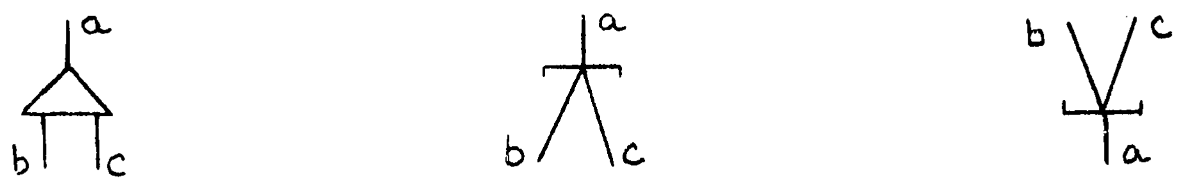
Notice that when a grammar is given in explicit form as in figure 2, each non-terminal, non-initial symbol has exactly two occurrences - once on the left side of a rule, and once on the right side. Thus a symbol on the right side of a rule serves as a pointer, or link, to the next rule to be executed, and a symbol on the left side of a rule serves as a pointer or link, back to the previous rule executed. A linked structure in which every forward link has a corresponding backward link is known as a symmetric list (Weizenbaum, 1963). Such a structure suggests that one might represent the information in the form of a graphical network in which the rules are nodes, and the symbols, which are really links, are lines connecting the nodes.

Let us produce such a graph for our sample grammar. First let us simplify the form of our grammar given in figure 2. We can eliminate all rules of the form $x \rightarrow y$ where x and y are single symbols without loss of information. We simply eliminate the rule, and replace the other occurrence of x with y. This simplification leads to figure 3. (We have renamed NP_1 SUBJ and have replaced NP_3 by OBJ. This

(3.1)	S	$\rightarrow$	SUBJ	VP
(3.2)	NP	$\rightarrow$	A	N
(3.3)	VP	$\rightarrow$	V	OBJ
(3.4)	A	$\rightarrow$	a,	the
(3.5)	N	$\rightarrow$	boy,	girl
(3.6)	V	$\rightarrow$	hit,	kissed
(3.7)	SUBJ, OBJ	$\rightarrow$	NP	

Figure 3: Explicit form of C-F grammar simplified

allows us to drop all subscripts.) We see that we now have in our grammar three types of rules - one type expressing concatenation (rules 3.1 through 3.3), one type expressing disjunction (rules 3.4 through 3.6), and one type expressing reverse disjunction (rule 3.7). We shall express each of these three types of rules with a different node, as shown in figure 4.[3] The obvious relationship between disjunction and reverse disjunction is expressed by using the same shaped node turned upside down. Using the appropriate node for each of the rules on figure 3, we connect the links as the rules instruct us, and we arrive at the network shown in figure 5.



(DOWNWARD ORDERED AND) concatenation

(DOWNWARD UNORDERED OR) disjunction

(UPWARD UNORDERED OR) reverse disjunction

Figure 4: The three nodes needed for C-F grammar

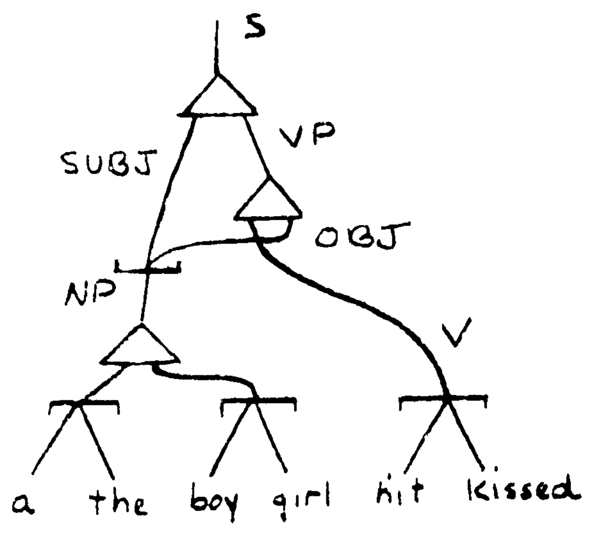


Figure 5: The grammar of figure 3 expressed graphically

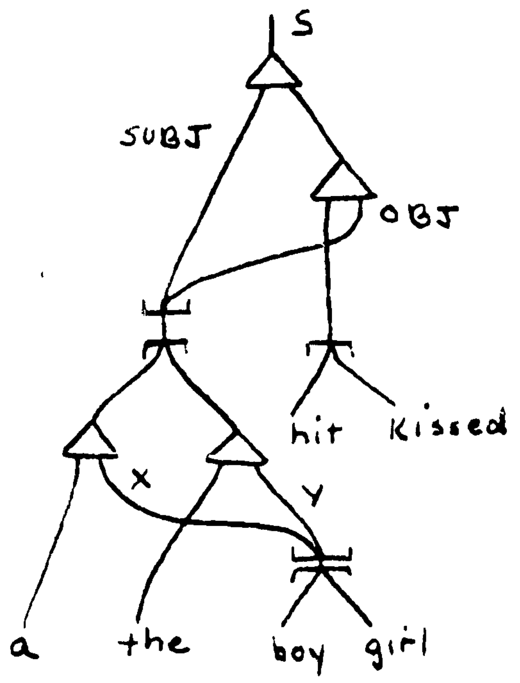
If we allow parenthesis notation in our algebraic formulas, by applying substitution to the seven rules of figure 3, we see that we can express the same grammar using only two rules, as shown in figure 6. We now have three sets of rules, all explicitly describing the same linguistic structure. In the set given in figure 3, each line of the

(6.1) S $\longrightarrow$ SUBJ ((hit, kissed) OBJ)

(6.2) SUBJ, OBJ $\longrightarrow$ ((a, the) (boy, girl))

Figure 6: Representation of grammar using parentheses.

network has a symbolic name in the algebraic form; in the version of figure 2, some lines have more than one name (for example, VP_1 and VP_2 name the same line); in the version of figure 6, some lines are not named at all. Is one version to be preferred to another? From the point of view of linguistic structure, the answer is no! They all describe the same structure. Figure 6 has fewer rules and fewer symbols, but that is just a notational artifact.[4] Consider the grammar of figure 7. It generates the same set of sentences as the grammar of figure 3.



$$S \rightarrow \text{SUBJ ((hit, kissed) OBJ)}$$
$$\text{SUBJ OBJ} \rightarrow \text{(a X), (the Y)}$$
$$X, Y \rightarrow \text{boy, girl}$$

Figure 7

It is described algebraically with the three rules using 31 symbols. Is it to be preferred to the grammar of figure 3, which has seven rules using 32 symbols? In order to answer this in a way that is linguistically interesting, one must go to networks which each of the algebraic formulations represent. If we compare the network in figure 5 to the network in figure 7, we find that the network in figure 7 is more complicated than that of figure 5. We see that in figure 5 the lines labelled a and the immediately come together in a disjunction. In other words, they both belong to the same class. This generalization is not made explicit in the structure of figure 7. Thus even using the trivial grammars of our examples we find counterexamples to the notion that complexity can be measured by counting symbols. A symbol count measures cleverness at algebraic manipulation rather than linguistic complexity. It would seem to be much better to base a complexity count on the network formulation, regardless of how it is described algebraically.

Two ways to measure the complexity of a network immediately come to mind. One is a count of the number of nodes and the other is a count of the number of lines. If all nodes have exactly three lines coming out of them, it does not matter which we choose, because the number of lines equals the number of nodes times 3/2 plus the number of lines which lead out from the network (i.e. connect to only one node) times 1/2. Since we only compare two grammars if they are equivalent in effective information - i.e. if they encode and decode the same set of structures (Lamb, 1966b:41-46) - the number of lines which lead out from two networks we are legitimately comparing will always be identical, and thus the one measure is a linear mapping of the other. Since we are only interested in complexity as an ordinal scale (McGinnis, 1965:274-290), the two measures would be equivalent. However, since we do not want to constrain our network to nodes with exactly three lines coming out of them, we must either use the count of the number of lines[5], or modify our node count to take into account the cases where more or less than three lines connect to a node. One way to take these cases into account is to count nodes connected by three lines as 1, and to add 1 to the count for each extra line. Thus a node connecting four lines would have a count of 2, a node connecting five lines would have a count of 3, and so on. A node connecting two lines would have a count of zero. These two counts do not give the same results in all cases. We shall return to this issue later.

Once we express a grammar in terms of a relational network, intermediate symbols become superfluous. What has become of the concept of the rewrite rule? We find that we can replace it with the notion of signals moving through the network. In the case of a grammar utilizing only the three nodes already discussed, the behavior of the signals moving throughout the network is relatively straightforward.[6] We start by sending a signal down from the top of the network. When the signal comes in the a wire of the concatenation node (see figure 4), the signal continues down the b wire. Thus in figure 5 a signal moving down from S would move down to SUBJ. When a signal comes in the b wire of reverse disjunction, it continues down the a wire. Thus the signal at SUBJ would reach NP, and by the instruction already given, would continue to A. When a signal comes in the a wire of disjunction, there are two possibilities. Either the signal may continue down the b wire, or it may continue down the c wire. If one is thinking in terms of generating sentences at random, each such choice can be made at random. If one is interested in the set of all sentences, each such choice doubles the number of potential sentences. Let us follow the c wire. This leads to the edge of the network labelled 'the', and we say that 'the' has been output from the grammar. Timing in this model is handled by feedback signalling, so a signal moves up from the line marked 'the'. A signal coming in on the c wire to a disjunction continues up the a wire. In this case the signal comes to A. A signal coming in the b wire of concatenation results in a signal being sent down the c wire. We have arrived at another disjunction. Let us say that this time the b wire is chosen. 'Boy' is output, and feedback is sent up to N. A signal coming in the c wire of concatenation continues up the a wire, in this case up to NP, and from there up to SUBJ, then down to VP, then to V, then a random selection, let us say to 'kissed'. Feedback travels up to V, then down to OBJ.

Here a further specification is needed. When a signal moves from OBJ down to NP, the node must 'remember' the fact, so that when feedback comes up to NP. the signal will return to OBJ rather than to SUBJ We say that as the signal passes through the reverse disjunction node from the c wire to the a wire, the node changes state, and stays in that state until a signal comes in from the b wire, whereupon the node returns to its original state.

Thus word by word, a sentence such as 'the boy kissed the girl' is produced. We have replaced the process of rewriting symbols by a process of moving signals through the networks. The instructions concerning how the signals move through the networks are given in terms of state-transition definitions of the nodes. There definitions are summarized in figure 8. As we build grammars to handle more complicated

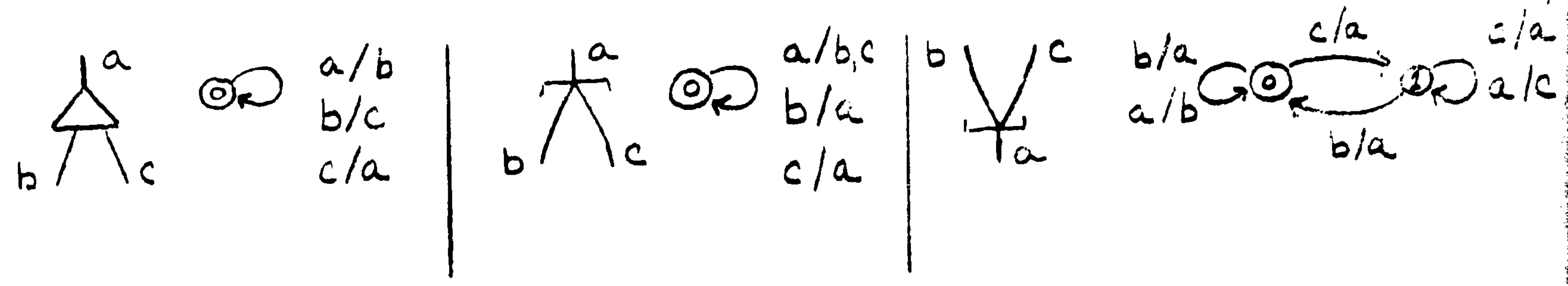


Figure 8: State transition defintions

linguistic information, these definitions become more complex and we find that we need many different types of nodes not yet discussed, but the basic idea of signals moving through networks remains unchanged One hypothesis we are exploring is that each of the nodes is finite We shall not explore this question further here, since it is covered in some detail elsewhere (Reich, 1968d).

Let us turn now to some formal properties of our relations. One such property is commutativity. A relation R is commutative if b R c is equivalent to c R b. In other words, one could replace b by c, and c by b in the algebraic expression without changing the relationship of b and c to each other. In terms of state transition definitions, the definition must be symmetric with respect to b and c. We see that this is true in the case of disjunction as defined in figure 8. In our diagrammatic notation we represent commutativity by connecting the b and c wires to the node at the same point. Thus it is always obvious at a glance whether or not a particular node is commutative.

Another formal property is associativity. A relation R is associative if b R (c R d) is equivalent to (b R c) R d. All the relations in our system are associative. This means that when two identical relations are connected to one another as shown in part (1) of each of the three triads of figure 9, the behavior of that portion of the network will

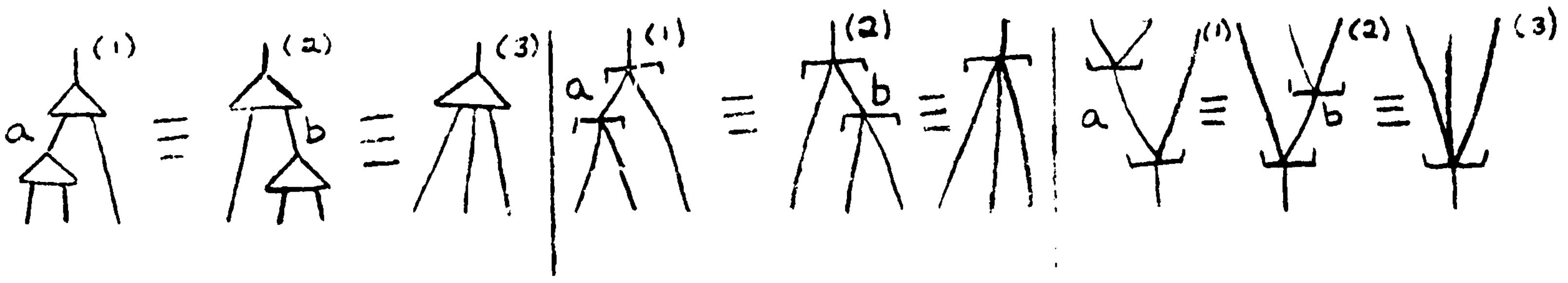


Figure 9: Associativity of relations

be exactly the same as seen from outside the construction as the corresponding relations in part (2), and vice versa. Since the choice of one over the other would be arbitrary, we express the construction as a three-way relation, as shown in parts (3) of the figure

The associativity requirement has a clear implication for the definition of processing in terms of signals moving through the network. No timing requirements can be based on the number of nodes the signal passes through, since the behavior of networks (1) is defined to be equivalent to the corresponding networks (2).

We have introduced the notion of an n-ary relation, where n is greater than 2. Do we allow arbitrarily large n-ary relations defined arbitrarily, or do we limit them? Our current policy, based on a desire for definitional simplicity (Reich, 1968a) is this: The only n-ary relations we define are those which arise from associativity of directly connected identical relations (such as those shown in figure 9), and these are defined to behave identically to an associative construction of binary relations. Thus we need only define binary and unary relations.

We also require, of course, that the grammar be finite, so that the number of wires leading from any one node must be finite. While this policy is based on formal considerations, is it possible to justify it on psychological or neurophysiological grounds? That is, is it possible that the formal principle may have some underlying psychological foundation? While it is highly speculative at this time, the possibility should not be overlooked. There is a principle in biochemistry that a molecule formed from three or more molecules is always formed in binary stages. The probability is too small that the three molecules will collide simultaneously in the proper orientation. Is it unreasonable to assume that the child learning his language builds up the underyling neural structure step by step in very small units? This intuitively appealing idea poses a problem for the developmental psycholinguist trying to analyze his data within a transformational framework, since transformations are intrinsically more complex. 'To discover the exact point at which a child's grammar contains transformational rules is a difficult, obscure problem,' writes David McNeill (1966:54). He characterizes the process as the building of a system of cumbersome and inelegant phrase structure rules, which are then junked in favor of transformations. 'The pressure - or, if you prefer, the motivation - to devise transformation rules may come from the cognitive clutter that results from not having them (1966:61)' We see that McNeill retreated from an incremental approach to the synoptic concept of 'cognitive clutter.'

To return to associativity, if concatenation is associative, what is the meaning of immediate constituent analysis or phrase structure parenthesization? Why do we prefer 10.1 over 10.2 (Lamb, 1966b:54).

(10.1) ((un true) ly)

(10.2) (un (true ly))

(10.3) (the (king (of England))) ((open ed) Parliament)

(10.4) (the (king of)) (England ((open ed) Parliament))

(10.5) (they (are (flying planes)))

(10.6) (they ((are flying) planes))

(10.7) (peasants (throughout China)) (work (very hard))

(10.8) ((peasants throughout) (China work)) (very hard)

Figure 10: Phrase structure parenthesization

10.3 over 10.4 (Wells, 1947:187-191), 10.7 over 10.8 (Nida, 1949:87), and why do we associate 10.5 with one interpretation of the sentence and 10.6 with another (Chomsky, 1956:118; Chomsky and Schützenberger, 1963:122)? The answer in terms of relational networks is quite simple. Two identical elements connected as shown in figure 9(1) can be reassociated to (2) if and only if there is no intervening element on line a. Similarly two identical elements connected as shown in (2) can be redistributed if and only if there is no intervening element on line b. Thus a particular phrase structure parenthesization means that the concatenation elements are separated by other nodes in such a way that no other associations are possible without making the structure more complex. We shall discuss this in more detail after we have concluded our discussion of formal properties.

A third formal property is distributivity. In arithmetic this means that (a + b)·c is equivalent to a·c + b·c. We say that multiplication distributes over addition. In our system, concatenation distributes over disjunction. This property is shown in figure 11.

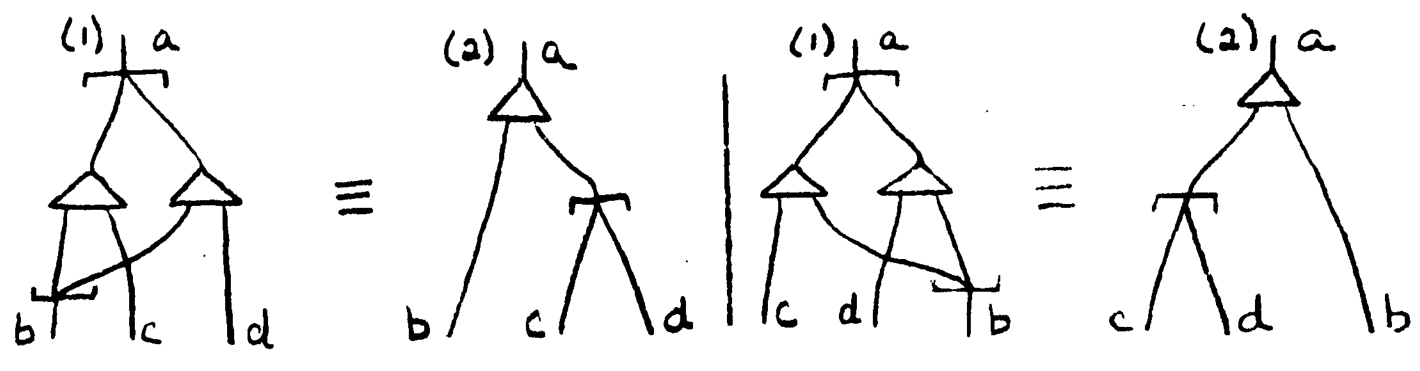


Figure 11: Distributivity

In (2) concatenation is expressed once for the set of elements (c,d). In (1) concatenation has been distributed over the disjunction. It is expressed once for each element in the set. Since concatenation is not commutative, we must state the distributive rule in two forms - left distributivity (figure 11, left), and right distributivity (figure 11, right). Notice that the difference between the grammar of figure 5 and that of figure 7 is that figure 7 is the result of distributing the concatenation element below NP in figure 5 over the disjunction below A.

The next formal property is coincidence. The lines leading down from a disjunction are said to be elements of that disjunctive set. Two sets are coincident if they contain the same elements. We see in figure 12 that one need never state the same set twice, as is done in (1).

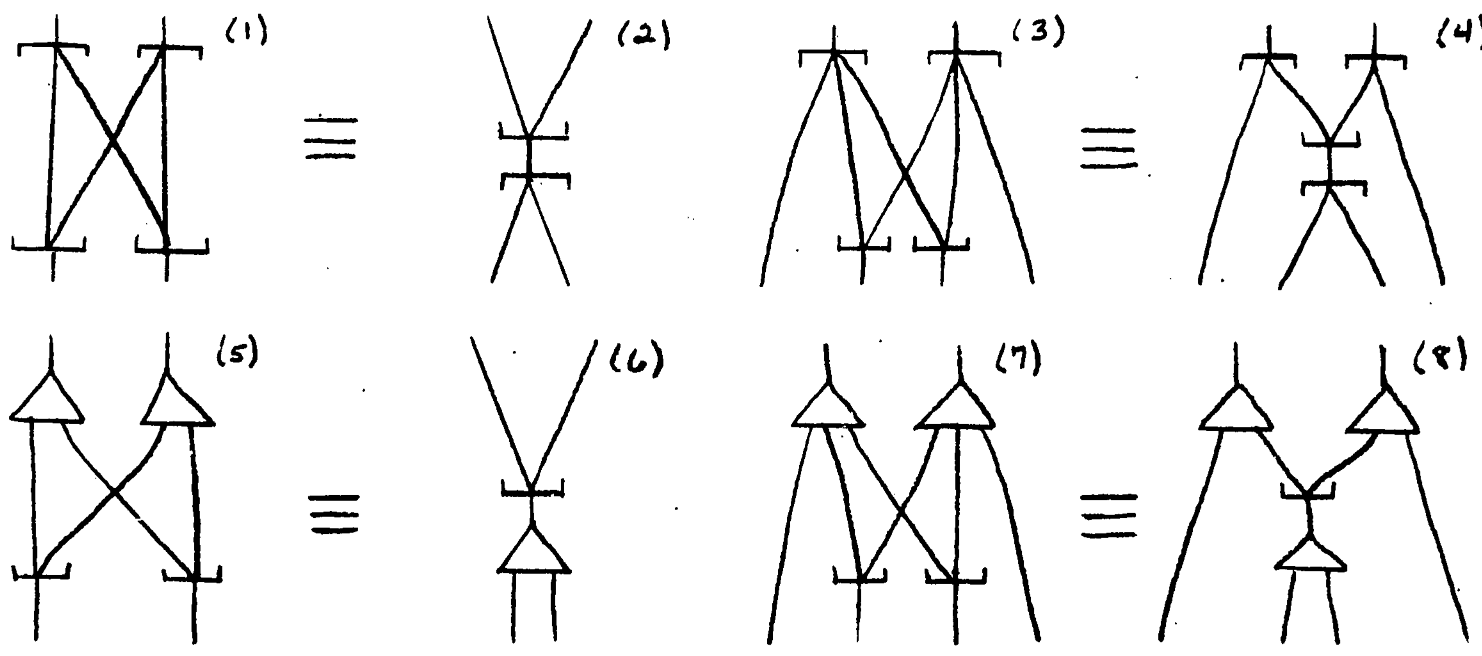


Figure 12: Coincidence

Rather, the same effective information can be expressed once as in (2), and the fact that the set is useful in two places in grammar is represented by reverse disjunction. The simplification of set overlap shown in (3) to structure (4) follows by application of associativity and/or commutivity combined with set coincidence. Concatenative coincidence, in which a particular concatenation is expressed twice, as in (5), can be simplified in the same way, as shown in (6). Similarly, concatenative overlap is the application of associativity (but not commutivity) combined with concatenative coincidence, converting structures like (7) to (8).

An additional, rather trivial, formal operation one can perform in simplifying networks is set reduction. If two elements of a set lead to identical structures, one of the elements is redundant, and can be eliminated. Thus in figure 13, (1) can be replaced by (2), and (3) can be replaced by (4).

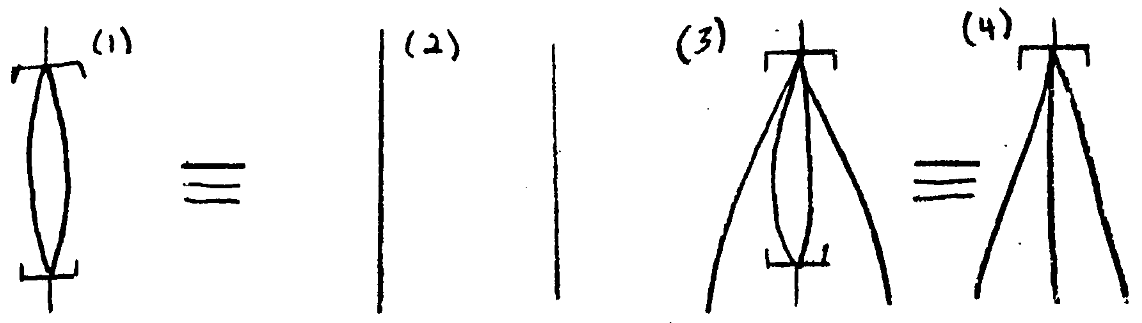


Figure 13: Set reduction

Let us now turn to the surprisingly thorny issue of optionality. Consider the case where a particular wire in linguistic structure is to be realized as either b c d or c d, as occurs, for example, in 'un friend ly' vs. 'friend ly'. Using the notation we have developed thus far, this can be expressed either as shown in figure 14(1) or as in 14(2). Given no other information, there is no reason to choose (1) over (2) or vice versa. One of our goals in constructing a formal theory is to develop a third way of expressing the structure, neutral with respect to the artificial distinction which differentiates the

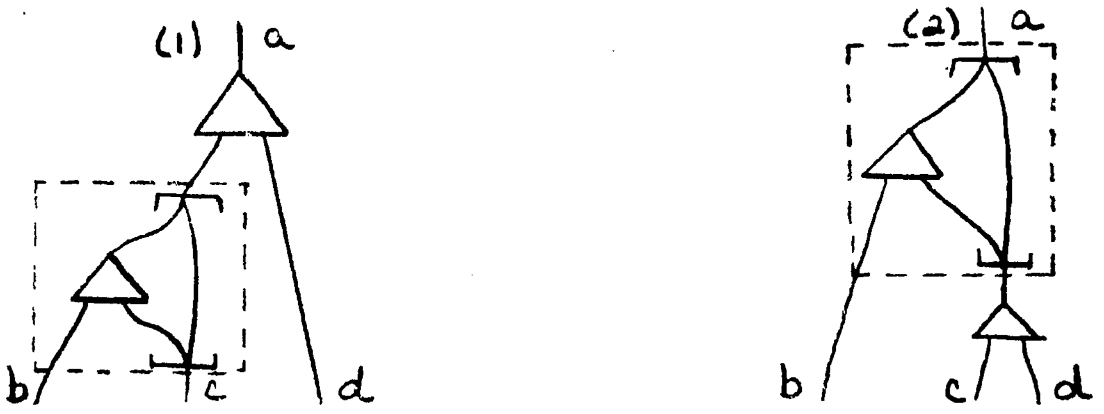


Figure 14

other two. When this question arose earlier in this paper, it was a case of associativity, and the problem was resolved by introduction of the n-ary relation. Can this problem be solved by reducing it to the already solved problem? The answer turns out to be yes. In (1) and (2), if that portion of the structure which appears within the rectangle of dashed lines is considered to be a single node, (1) is derivable from (2) by reassociating the two nodes. Do we need to consider this a case of associativity of two different types of nodes? No. Consider the alternate formalism of indicating the optionality by putting an 'optional' element on that line. If the optional element is a small circle, we arrive at 15(1). Notice that 15(1) has an alternate description in 15(2), but since we now have two contiguous concatenation

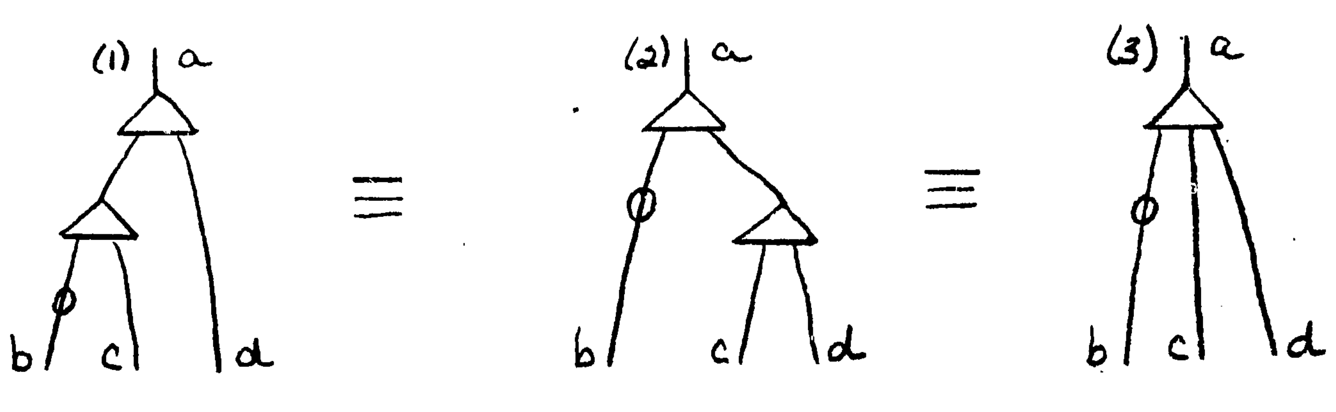


Figure 15

elements, we can apply our already established associativity rule to arrive at 15(3). This expresses the same relationships, while being neutral with respect to the artificial distinction. In Lamb 1966b, optionality was expressed by using a zero element attached to a disjunction. However, considerations resulting from experiments with various models of performance indicate that this representation is inappropriate, and that it is best to represent optionality as a separate element. There is one thing wrong with this formalism. The particular case in which wire a is realizable as b, or c, or b c, but not ∅ (nothing), cannot be represented in this notation. One must return to the awkward format of a bypassing line, as shown in figure 16(1). The structure shown in 16(4) allows b, c, or b c, but also ∅.

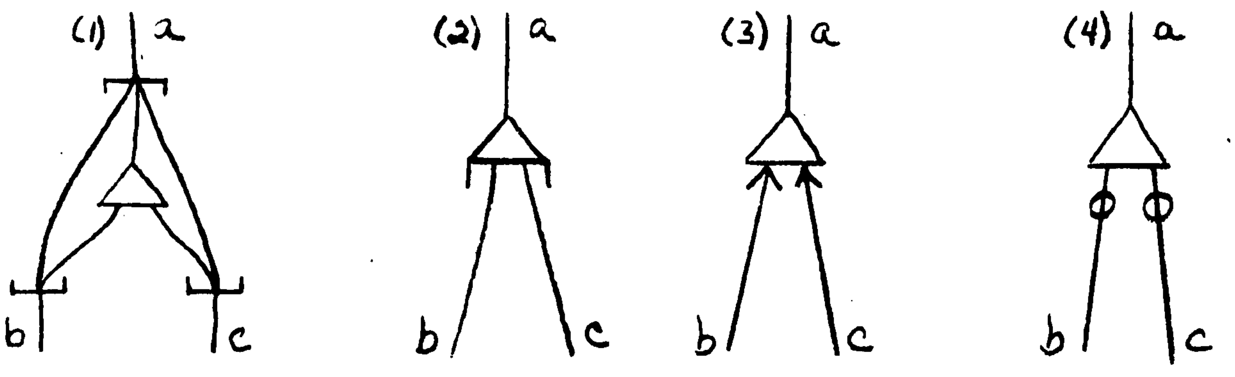


Figure 16

One solution is simply to define a new node to fill the gap. This is the and-or, shown in 16(2), and defined to be identical to 16(1). It would be nice if we could get along without inventing a new node. One attempt was the arrowhead notation used in Sampson (forthcoming). An arrowhead on a line meant that that line was independent, and could essentially bypass the concatenation element immediately above it.

Thus in figure 17(1) a could be realized as b c d or c d. This notation has the advantage that an arrowhead on both lines leading down from a concatenation element, as shown in 16(3), would have the same meaning as our missing case, 16(1), namely, that b can be an independent realization, that c can be an independent realization, or that b c can be a realization. In order to allow the possibility

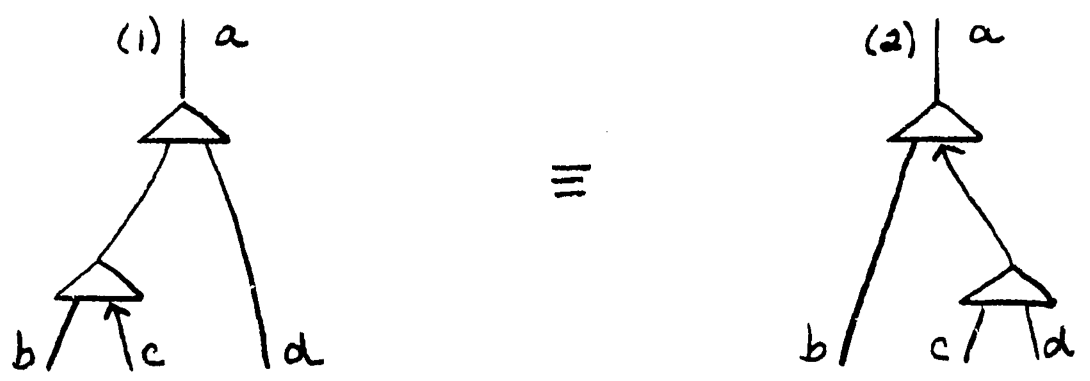


Figure 17

of nothing as a realization, one would have to put another arrowhead on the next concatenation element up, as shown in 18(1). If reverse disjunction occurs first, this fact would have to be expressed more than once, as shown in 18(2). Worse yet, this notation has exactly the same fault as the original bypassing line notation. Figure 17(1) is equivalent to figure 17(2), and there is no way to neutralize the artificial distinction between the two.

I suggested another possibility - simply define the optional elements so that when one occurs on both lines leading down from concatenation, as in 19(1), its meaning is by definition equivalent to the missing case, namely 19(2). In order to allow the option of realization as nothing, one puts an optional element on the wire

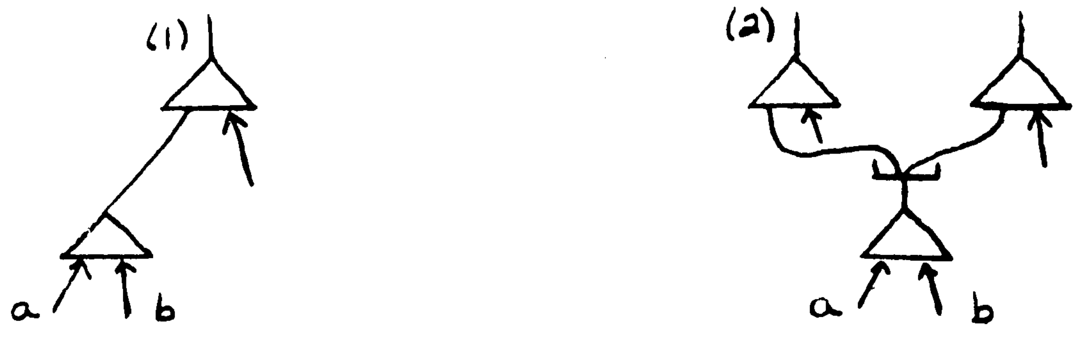


Figure 18

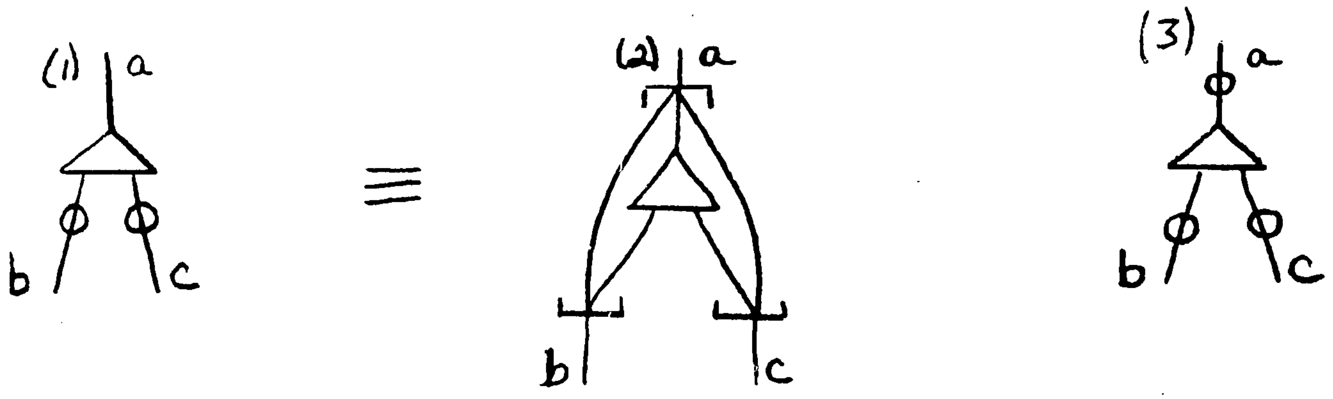


Figure 19

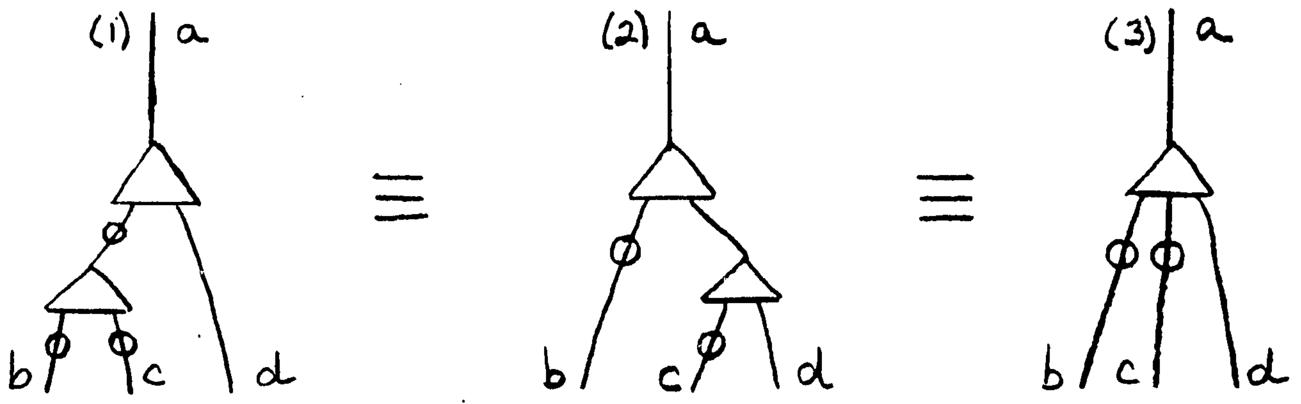


Figure 20

leading up from the concatenation element, as shown in 19(3). This neutralizes the artificial distinction between 20(1) and 20(2) by the representation 20(3). Unfortunately, this solution is not without problems. 20(1) looks more complicated than 20(2) since it requires one more optionality element. But this is purely a notational artifact. The two concatenation elements in 20(1) can be associated, in spite of the fact that there is an element on the line between them. Thus we would have an exception to our rule about when to apply associativity. Worse yet is the exception to the associativity applicability rule that would result from figure 21. This structure would

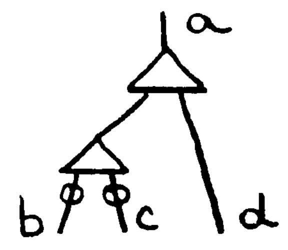


Figure 21

allow b d, or c d, or b c d. In spite of the fact that the two concatenation elements are contiguous, they cannot be collapsed to a three way concatenation, nor is there an equivalent structure in which the association goes the other way.

Because of these problems, I have reluctantly concluded that the best solution to the problem of optionality is the one that was originally suggested; namely, mark the optional line with an optionality element, and introduce a new element, the and-or, to take care of the missing case.[8] Thus the structure represented in 20(1) would be represented by 22(1), and the associativity can once again be expressed in a straightforward way, 22(1) and 22(2) being equivalent to the

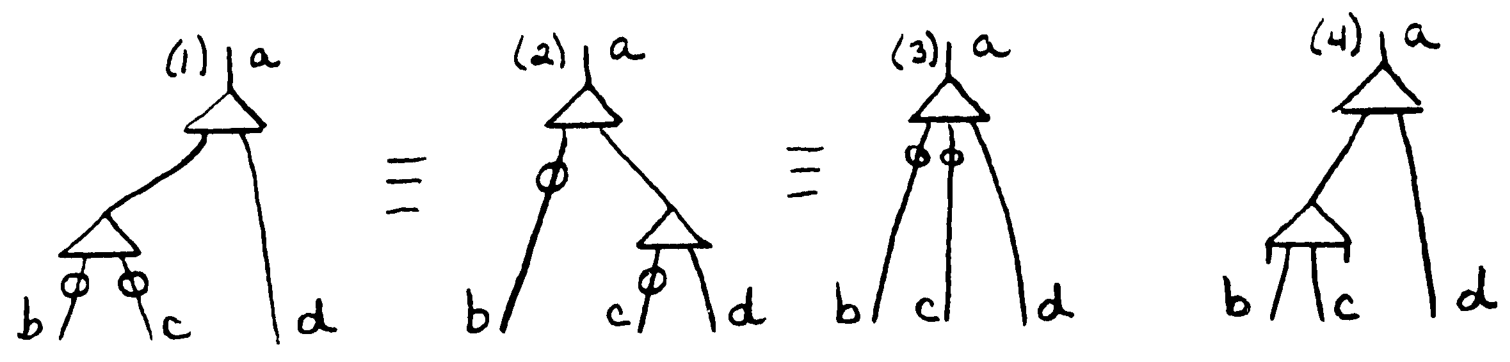


Figure 22

preferred 22(3). The problem structure figure 21 is represented using the new node, as shown in 22(4), and there is no problem. The <u>and-or</u> is not associative with concatenation, and that is that. The formal properties of the new node are similar to those of concatenation. Equivalence of effective information is preserved under associativity, right and left distributivity, and coincidence. The optional element shows equivalence under the properties of bypass representation, distributivity, coincidence, and reduction. The formal properties of all nodes thus far discussed are summarized in figure 23.

What is gained by this somewhat involved discussion of formal properties of nodes? Formalization is not a goal in itself, but is only meaningful if it leads to insights into the data being studied. Our goal in doing this is to increase the relevance of stratificational theory to the problems of grammatical discovery and language acquisition. The next section of the paper is devoted to a discussion of how we make use of the formal properties of nodes.

In our discussions of grammatical discovery, it seems useful to distinguish among at least three logically distinct operations.

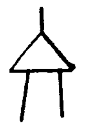
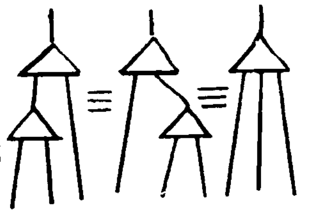
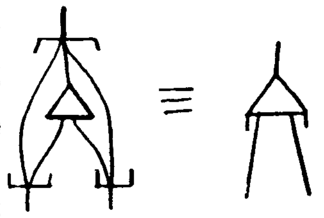
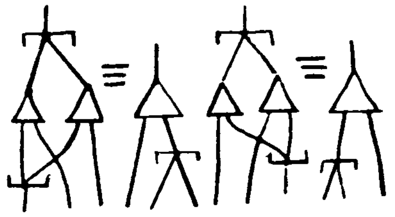
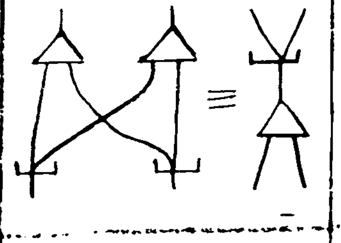
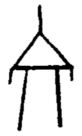
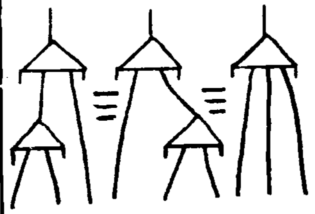
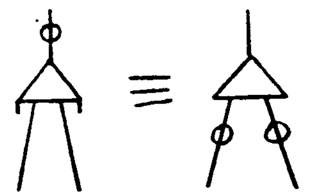
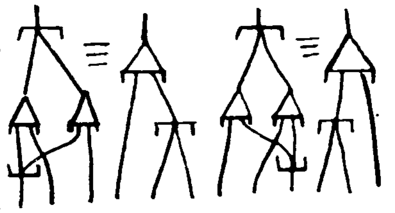
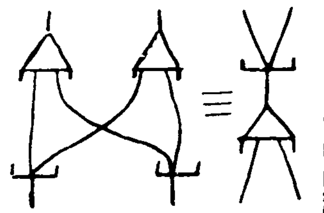
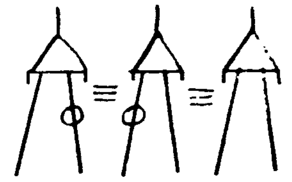
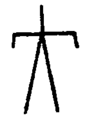
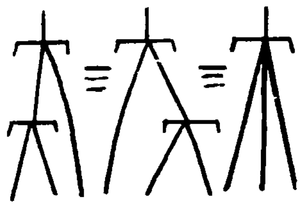
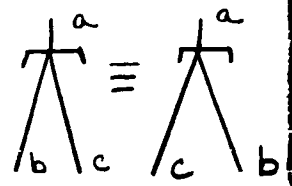
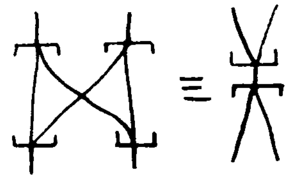
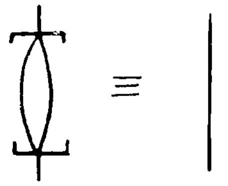
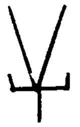
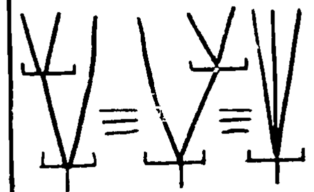
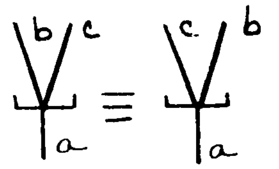
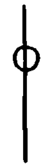
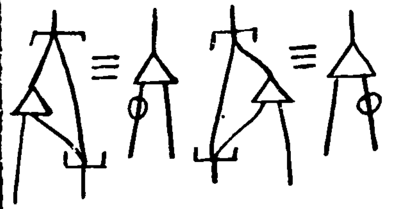
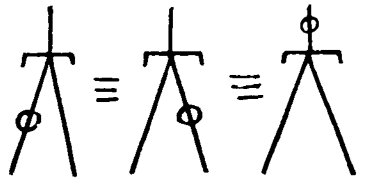
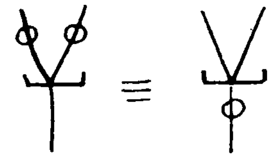
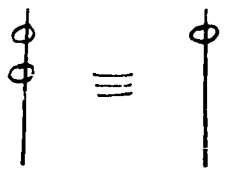
SYMBOL	NAME	A ASSOCIATIVITY	B BYPASS REPRESENTATION	C COMMUTIVITY	D DISTRIBUTIVITY	E COINCIDENCE	R REDUCTION
DCT 	DOWNWARD CONCATENATION (DOWNWARD ORDERED AND)			NO			—
DOA 	DOWNWARD ORDERED AND-OR			NO			
DDJ 	DOWNWARD DISJUNCTION (DOWNWARD UNORDERED OR)		—		EQUIVALENT TO ASSOCIATIVITY COMBINED WITH REDUCTION		
UDJ 	UPWARD DISJUNCTION (UPWARD UNORDERED OR)		—		—	—	—
DOP 	DOWNWARD OPTIONAL	—		—			

FIGURE 23: SUMMARY OF FORMAL PROPERTIES

They are association, simplification, and extrapolation. Association may be thought of as being either unimodal or bimodal. An example of unimodal association would be the child's learning that b followed by ɔ followed by l was a word in English, ball. Bimodal association would be associating the word ball with an object the child is looking at, or with a set of objects the child knows about. Given a grammar which describes the set of utterances the child has heard and understood, extrapolation would be the process of producing a grammar which goes beyond the data, in the sense that it is capable of producing not only the data already given, but also utterances that the child might reasonably expect to run across in the future. In order for association and extrapolation to be possible, a process of simplification must also occur.

In this section we shall concern ourselves exclusively with simplification. We shall use as an example Lamb's exercise of determining the constituent structure of (un true ly) (1966:54). Considered by itself, of course, we have no reason to prefer one association over the other. Therefore we look at related data from English. Let us consider first the four adverbs (un true ly), (un wise ly), (true ly), and (wise ly). We say that they all occur in English, which we can express in network terms by simply listing them as members of the disjunctive set of adverbs, as shown in figure 24(1). We shall call this form the disjunctive form of a network. The grammar which consists of merely listing the data is, of course, trivial and uninteresting. However, by utilizing the formal properties given in figure 23, we can convert a network to other networks which are equivalent

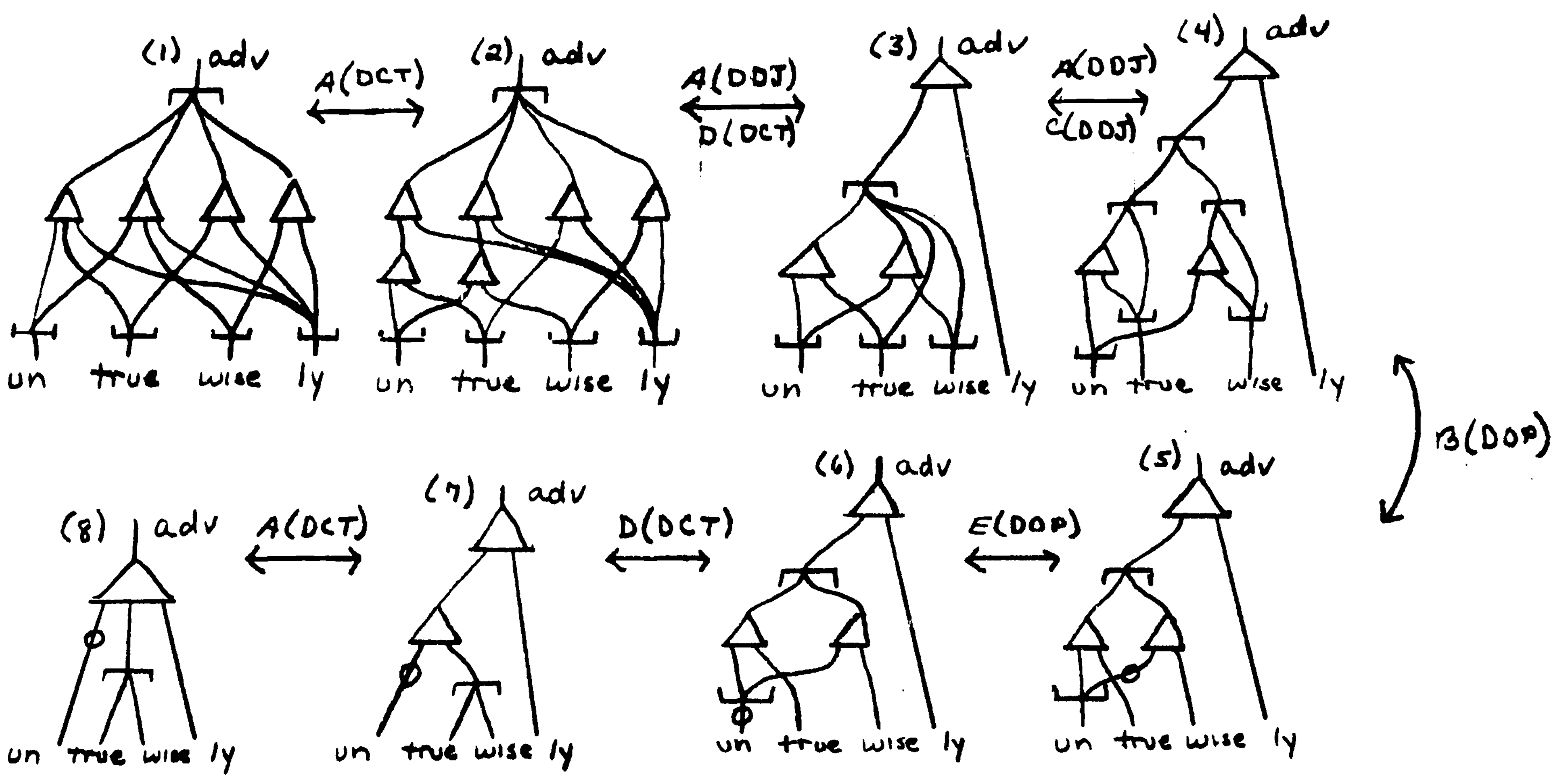


Figure 24

in terms of effective information. Our goal is to convert the given network to the simplest possible network which is equivalent in effective information. We see that by application of associativity of downward concatenation (A(DCT)) twice to 24(1) we arrive at 24(2). Then by applying associativity of downward disjunction (A(DDJ)) and distributivity of downward concatenation (D(DCT)) we arrive at 24(3), and so on, step by step until we arrive at 24(8), which is the simplest representation. We see that given only the information about the four adverbs, there is no reason to parenthesize (un true ly) any further. We therefore consider some additional information about English; namely, that it contains the four adjectives (true), (un true), (wise), and (un wise). In other words, we know that a structure equivalent to 25(1) is also a part of the structure of English. First we

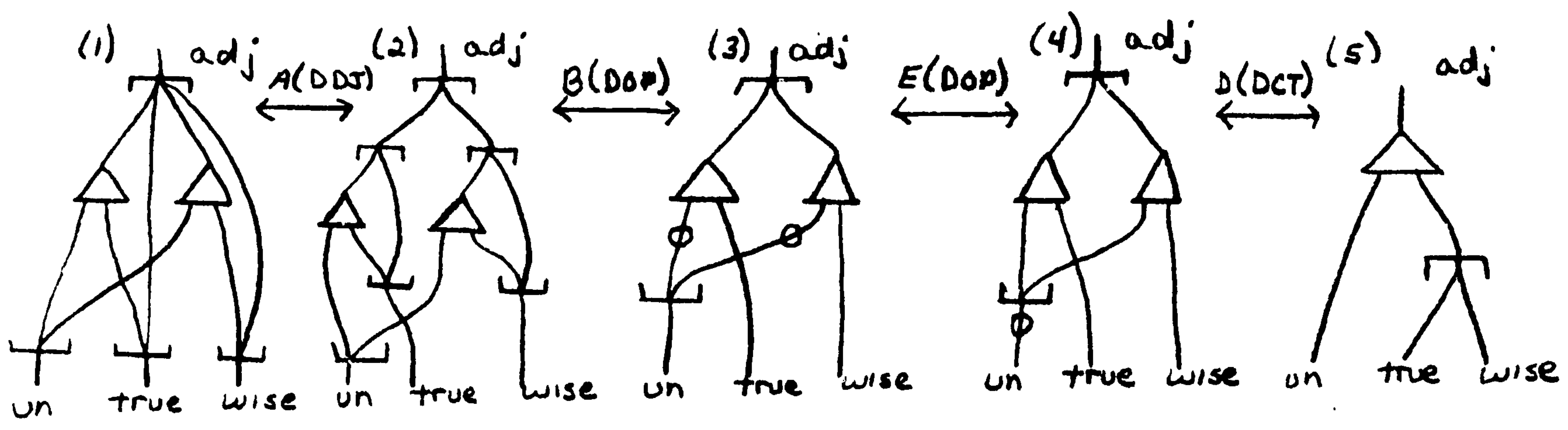


Figure 25

simplify 25(1) by making use of the appropriate formal properties, as shown in the figure, finally arriving at 25(5). Next we combine the adverb structure of 24(8) with the adjective structure of 25(5), resulting in the structure of 26(1). Utilizing the formal properties

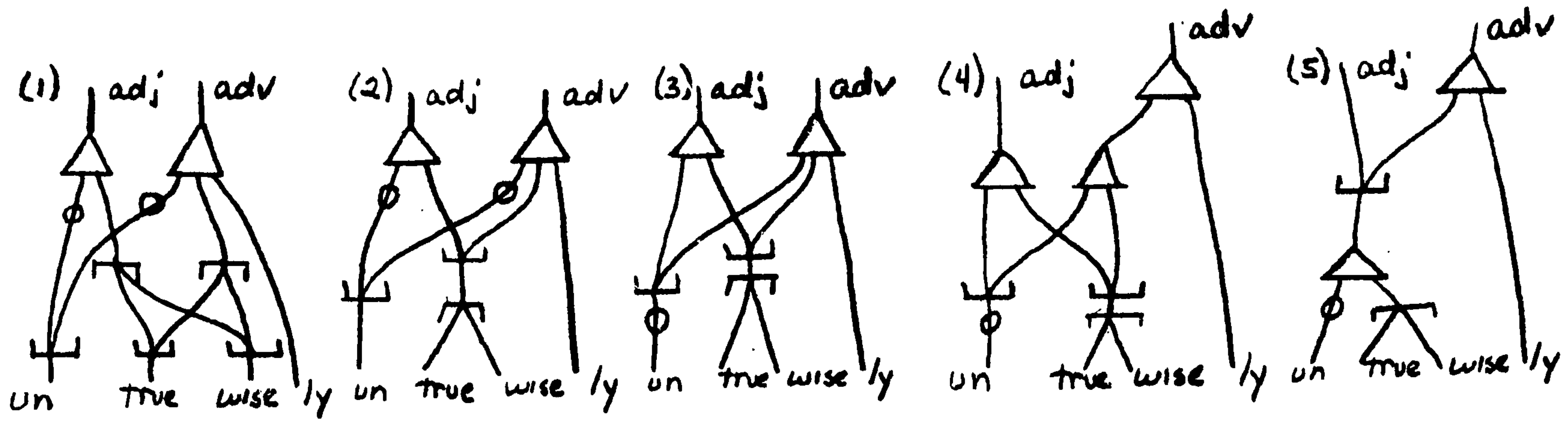


Figure 26

again, we discover we can reduce the network to 26(5) without affecting the information in the network. We see that in our final network there is a node between the two concatenation elements, so that one constituent analysis is to be preferred, namely ((un true) ly).

Let us step back and take a look at what we are doing. Consider the set of all possible networks. We can partition this set into a set of mutually exclusive subsets such that two networks are in the same subset if and only if they are equivalent in effective information. Let us look at one particular subset, say the subset of all networks which are equivalent to 24(1). If we think of each network as a point, we can consider the equivalences stated as formal properties as lines connecting points. One of my goals is to identify enough equivalence relations such that all points in the subset (all equivalent networks) are connected to one another so that there is at least one path of logical operations from any one network to any other. The value of such a goal should be obvious. Consider a computer program whose task is to find the simplest network equivalent to 24(1). One way it could go about its task would be to generate, one by one, the set of all possible networks with complexity less than 24(1), then test each one to determine if it is equivalent to 24(1). It would be much easier if the machine had a way of generating all equivalent networks directly, which it would have if it had a complete set of equivalence relations.

Let us assume that we have succeeded in connecting all equivalent networks by means of a network of equivalence relations. What can we say about a procedure for finding the simplest network?[9] Consider the points representing networks to be located in a plane. Now consider the number which we use as a measure of complexity to represent a height above the plane immediately above each point. A surface could be made to pass over each of the points at the height given by the measure of complexity. We might picture this surface as hilly countryside. Our problem, stated in these terms, is that, starting from a given point on this countryside, we must explore the countryside by following the given paths, searching for the deepest valley (lowest complexity count). One technique is, of course, exhaustive search, in which all points are looked at, after which the least complex is chosen. This technique is almost never reasonable, and our problem is no exception. Each of our equivalence subsets contains an infinite number of points, which would mean the search would take a very long time indeed. Another technique is to move down the paths in such a way that you are almost always going downhill. That is, at almost every step you take, the complexity count afterward is less than it was before. This technique is known as hill climbing (Minsky, 1961:409-411). The successfulness of this technique depends upon the ruggedness of the problem terrain. If there is more than one isolated valley, then our walk downward may take us to the wrong place, from which vantage point it will not be at all obvious that there is an alternate, better solution. The ruggedness of the problem terrain depends importantly on the complexity measure used. We have considered two measures in this paper - a count of the number of lines, and a count of the number of nodes plus extra lines. Let us look at the ruggedness of the terrain in the equivalence space of figure 24 using each of the two complexity counts. In the case of the line count, the step from (2) to (3) is downhill, the step from (3) to (4) is uphill, and the rest of the steps from (4) to (8) are downhill all the way. In the case of the node plus extra line count, the step from (1) to (2) is level (no change in the complexity count), the step from (2) to (3) is downhill, the step from (3) to (4) is level, (4) to (5) is downhill, (5) to (6) is level, (6) to (7) is downhill, and (7) to (8) is level. If we try out our two measures on an alternate simplification path for the same example, shown in figure 27, the result is the same. Similarly, the result is the same in the different

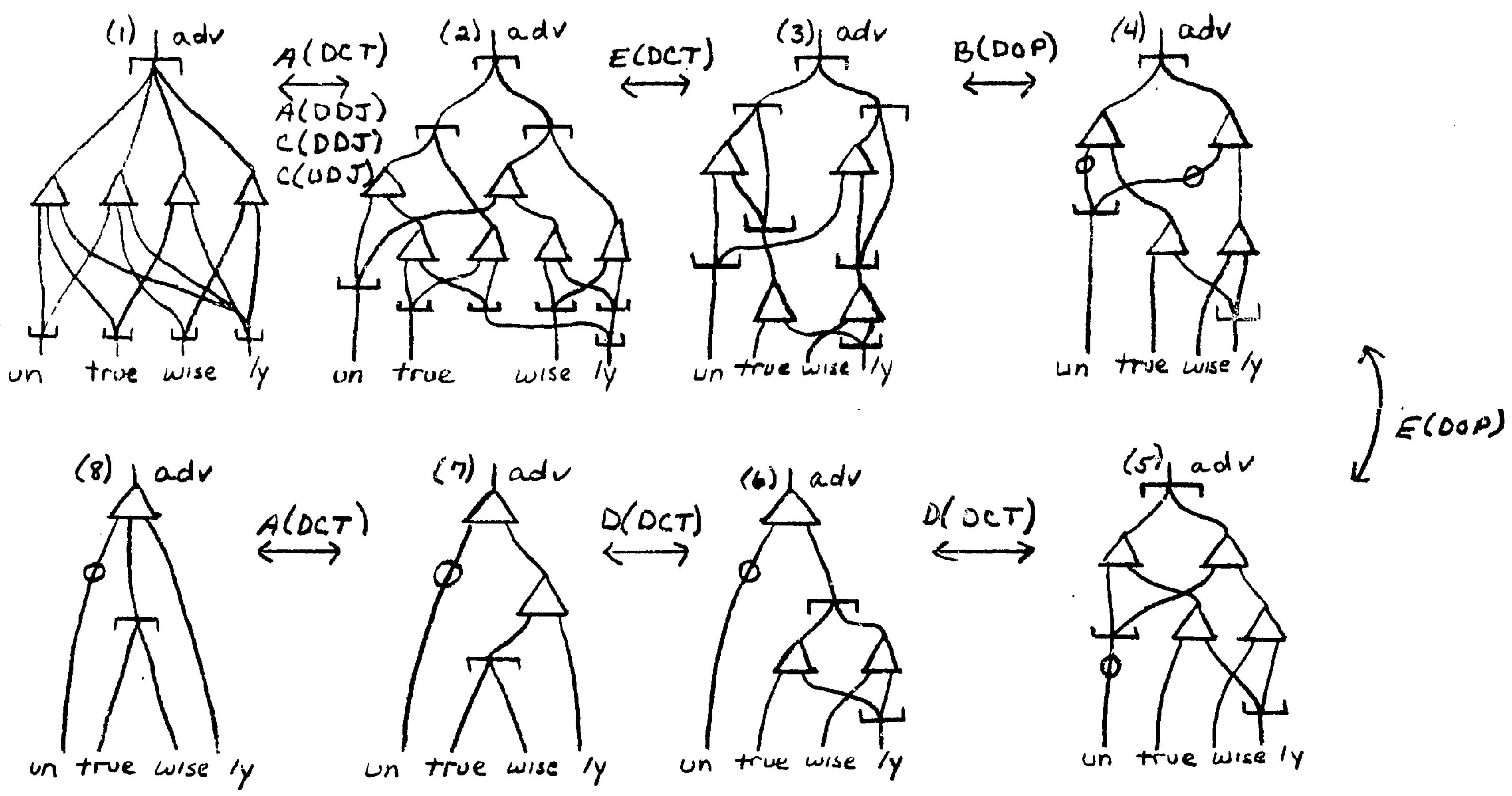


Figure 27

problem spaces represented in figures 25 and 26. Based on these examples, we would prefer the node plus extra line count, since it leads to a less rugged problem space. Before we eliminate the line count from our consideration, we should explore the difference between the two counts more fully. The two measures are strongly nonequivalent, which means that there exist pairs of equivalent constructions for which the two measures give opposing results. One such example is given in figure 28. The line count prefers 28(2) at 8 lines to 28(1) at 9 lines, while the node plus extra line count prefers 28(1) at 4 to 28(2) at 6. The node plus extra line count corresponds to our intuition that 28(1) is the simpler. Examples such as these give us additional reasons to prefer the node plus extra line count. However, we do prefer figure 24(8) to 24(7) and 27(8) to 27(7), which the line count expresses while the node plus extra line count is neutral. While the line count does handle associativity preferences, it fails to handle other preferences, such as distributivity of downward optional (figure 23). The diagram on the right is preferred to the other two, since it neutralizes the artifactual distinction between the other two structures. In figure 23 the preferred structure of each of the equivalent parts or triples is the one on the right.

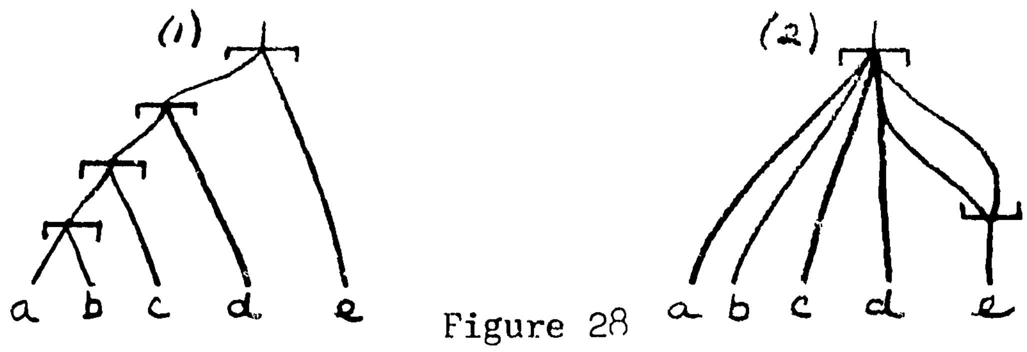


Figure 28

I have found no single number which selects all of the finer distinctions shown in the figure. Therefore I suggest the following two-part algorithm to arrive at the preferred diagram: Starting at the disjunctive form, hill climb by applying the equivalence relations utilizing the node plus extra line count as the measure of complexity. At any step, where more than one possible simplification is possible, each must be tried. When no further simplifications can be made, the simplest structure should then be fine tuned by applying equivalences which do not affect the complexity count, so that all parts of the final structure correspond to that shown on the right in each of the equivalences shown in figure 23. These equivalences are all local in nature, and thus can be applied in any order, independent of one another. The resulting diagram will be the simplest possible structure. This will be true if and only if the following conjecture is true: Given the node plus extra line count as the measure of complexity, there exists a sequence of equivalence operations which convert a network in disjunctive form to the simplest network which describes the same effective information such that the complexity of each step is less than or equal to the complexity of the preceeding step.

What this conjecture implies is that one may not need to postulate complex special purpose mechanisms to explain how children can learn language so rapidly. Instead, it suggests that the linguistic system may, if formulated properly, be much simpler than most theoreticians have realized, such that relatively simple learning mechanisms suffice to accomplish the seemingly herculean task of discovering the structure of language. This conjecture would be considerably stronger if one could state that no matter what sequence of equivalence operations was performed on the disjunctive form of a network, the resulting network would ultimately be in its maximally simplest form. To put it another way, it would be nice if all paths down the mountain lead to the same valley. While this seems to be true in the equivalence space used in figures 24 and 26, it is in general not true with the notation we are currently using. This does not preclude the possibility from being true in some other notational system. Although a heuristic program which chose an arbitrary sequence of simplifying equivalence operations would not necessarily end up with an optimal solution, this does not mean that it would not be a reasonable model of the simplification part of language acquisition. One need not assume that each child learning his native language necessarily finds the optimal solution. This area is discussed further in Reich, 1968 forthcoming.

We have tried to show in this paper that if a grammar is stated completely explicitly, it can be represented in the form of a relational network of the type proposed by Lamb. We have discussed some formal properties of such networks, and have made some revisions to Lamb's formulation which allow us to keep the formal properties and the structural complexity count as simple as possible. We have introduced the notion of hill-climbing on equivalence spaces defined by the formal properties and the complexity count as a model of the simplification part of language acquisition. We have suggested that using the complexity count we proposed, the process of simplification of a grammar in disjunctive form is non-uphill all the way. However, we have done all these things only with reference to a system of relationships equivalent to a context-free phrase structure grammar. In other papers in this series we extend these results to networks capable of handling those features of language which cannot be handled by context-free phrase structure grammars. Thus we find that the network approach not only gives a simpler overall system, but one which has the added benefit of starting us along the path of developing a detailed model of the process of language acquisition.

FOOTNOTES

1 Part of the misunderstanding probably results from the fact that Lamb is continually changing his theory, so that it is difficult to obtain a coherent picture by reading articles written several years apart. In order to alleviate this problem, Ilah Fleming is currently writing an annotated bibliography of the stratificational literature.

2 At least not until recently. In Chomsky (1966:424) a rule of this type is used. Like regular disjunction, this type of rule has the status of an abbreviation. As far as I can tell, Chomsky has not yet realized the full import of adding this relation to his system.

3 Throughout this paper I have tried to use Chomsky's terminology wherever possible, on the assumption that more readers have read and understood Chomsky than have read and understood Lamb. In the diagrams, Lamb's terminology is indicated in parentheses. The change in terminology is only an attempt to increase understanding. It has no theoretical significance.

4 Lamb (1966b:54-56) refers to this as superficial information.

5 This was suggested by Lamb (1966a:555) early in 1966. Later Lamb (1966b:46-54) went to a more complicated measure, because the measure gave incorrect results with respect to his treatment of optionality. In June 1966, I discovered a counterexample to this measure and pro-

posed still another measure, the node plus extra line count which has been in use up to now (Lockwood, 1967; Sampson, 1967). Independent considerations have led us to modify our treatment of optionality, and since the new treatment negated Lamb's 1966 objections to the line count, we shall reconsider it here.

6 The highly simplified model described in this section is not adequate to handle looping, which transformationalists refer to as recursion. Because of this, the simplified signal model is not equivalent to a context-free phrase structure grammar. If one allows push-down storage of states of the nodes, one can build a system that is strictly equivalent to such a grammar. As this topic is covered in detail in Reich, 1968d, it will not be discussed here.

7 One can criticize McNeill's proposal on other grounds. The proposal assumes that the concept of a transformation has some psychological validity. Recent experiments have raised serious doubts about this assumption (Fodor and Garrett, 1966; Reich, 1968a). McNeill's measure of cognitive grammatical structure in terms of a count of the number of rules is also questionable. For a more careful approach to psychological complexity within the transformational framework, see Brown and Hanlon (1968).

8 Gleason (personal communication) has suggested that concatenation, and-or, and disjunction can all be thought of as the same function of two variables - the minimum number of successful downward outputs, and the maximum number of same. In concatenation the min and the max

are both 2. In and-or, the min is 1 and the max is 2. In disjunction the min and max are both 1. This is an insightful way of understanding the nodes. However, I prefer to represent the nodes with different symbols, to emphasize the differences in their formal properties.

[9] This problem looks suspiciously like some problems for solving which mathematicians have proved there exists no algorithm (Trakhtenbrot, 1963:92-101). This sort of proof need not concern us, since it refers to an algorithm for solving the general class of problems, rather than any particular problem.

BIBLIOGRAPHY

Bach, E. (1964). An Introduction to Transformational Grammars. New York: Holt, Rinehart and Winston.

Brown, R. and Honlon C. (1968). Derivational complexity and the order of acquisition in child speech. Paper presented at the 1968 Carnegie-Mellon Symposium on Cognitive Psychology.

Chomsky, N. (1956). Three models for the description of language. I.R.E. Transactions on Information Theory. IT-2. 113-124. Page references refer to this version. (Also in Luce, R.D., Bush, R., and Galanter, E. (1965). Readings in Mathematical Psychology. vol. II, New York: Wiley.)

Chomsky, N. (1957). Syntactic Structures. (Janua Lingorum, no. 4) The Hague: Mouton.

Chomsky, N. (1963a). Introduction to the formal analysis of natural languages. In Luce, R. D., Bush, R. R., and Galanter, E. (eds). (1963). Handbook of Mathematical Psychology, vol II. 269-322. New York: John Wiley and Sons.

Chomsky, N. (1963b). Formal properties of grammars. In Luce, R. D., Bush, R. R., and Galanter, E. (eds). (1963). Handbook of Mathematical Psychology. vol II. 323-418. New York: John Wiley and Sons.

Chomsky, N. (1965). Aspects of the Theory of Syntax. Cambridge. Mass: MIT press.

Chomsky, N. (1967). Some general properties of phonological rules. Language. 43. 102-128.

Chomsky, N. and Halle, M. (1968). The Sound Pattern of English New York: Harper and Row.

Chomsky, N. and Schützenburger, M. P. (1963). The algebraic theory of context free languages. In Braffort, P. and Hirschberg, D. (eds). Computer Programming and Formal Systems. 118-161. Studies in Logic Series. Amsterdam: North Holland.

Fodor, J. and Garrett, M. (1966). Some reflections on competence and performance. In Lyons, J. and Wales, R. J. (eds). Psycholinguistic Papers. Edinburgh: Edinburgh University Press.

Gleason, H. A. (1964). The organization of language. In Stuart, C. I. J. M. Report of the Fifteenth Annual Table Meeting on Linguistics and Language Studies. 75-95. Washington, D. C.: Georgetown University Press.

Householder, F. W. (1965). On some recent claims in phonological theory. Journal of Linguistics. 1. 13-34.

Lamb, S. M. (1966a). Prolegomena to a theory of phonology. Language 42. 536-573.

Lamb, S. M. (1966b). Outline of Stratificational Grammar. Washington, D. C.: Georgetown University Press.

Lockwood, D. G. (1967). Some morphotactic properties of the Czech noun declension. Paper read at the Summer Meeting of the Linguistic Society of America.

Markov, A. A. (1954). Theory of Algorithms. Works of the V. A. Steklov Mathematical Institute. 42. Moscow, USSR: Academy of Sciences. (Translated by Schoor-Kon, J. J. and staff. Jerusalem, Israel Program for Scientific Translation. 1961. Washington, D. C.: Office of Technical Services, U. S. Department of Commerce. OTS 60-51085).

McGinnis, R. (1965). Mathematical Foundations for Social Analysis. Indianapolis, Indiana: Bobbs-Merrill.

McNeill, D. (1966). Developmental psycholinguistics. In Smith, F. and Miller, G. A. (eds). The Genesis of Language: A Psycholinguistic Approach. Cambridge, Mass.: MIT Press.

Minsky, M. (1961). Steps toward artifical intelligence Proceedings of the Institute of Radio Engineers. 49, 8 - 30. (Reprinted in Feigenbaum, E. A. and Feldman, J. (1963). Computers and Thought. New York: McGraw-Hill. 406-450. Page references refer to this version.)

Nida, E. A. (1949). Morphology: The Descriptive Analysis of Words. Ann Arbor, Mich.: University of Michigan Press.

Postal, P. M. 1968. Aspects of Phonological Theory. New York: Harper and Row.

Reich, P. A. (1968a) Competence, performance, and relational networks. Paper given at the December, 1968 Linguistic Society of America meeting. Linguistic Automation Project Report. New Haven, Ct.: Yale University.

Reich, P. A. (1968b) The relational network simulator. Linguistic Automation Project Report. New Haven, Ct.: Yale University.

Reich, P. A. (1968c). The English Auxiliaries: a relational network description. Linguistic Automation Project report. New Haven, Ct.: Yale University.

Reich, P. A. (1968d). The finiteness of natural language. Linguistic Automation Project Report. New Haven, Ct.: Yale University.

Reich, P. A. (forthcoming). Toward a model of language acquisition. Linguistic Automation Project report. New Haven, Ct.: Yale University.

Sampson, G. R. (1967). A stratificational analysis of the English numeral system. Paper read at the Summer Meeting of the Linguistic Society of America. Linguistic Automation Project Report. New Yaven, Ct.: Yale University.

Trakhtenbrot, B. A. (1963). _Algorithms and Automatic Computing Machines._ Boston: D. C. Heath and Co.

Wells, R. S. (1947). Immediate constituents. _Language_. _23_. 81-117. (Also in Joos, M. (1957). _Readings in Linguistics_. Washington, D. C.: American Council of Learned Societies. 186-297. Page references refer to this version.)

Weizenbaum, J. (1963). Symmetric list processor. _Communications of the ACM_. _6_. 524-543.